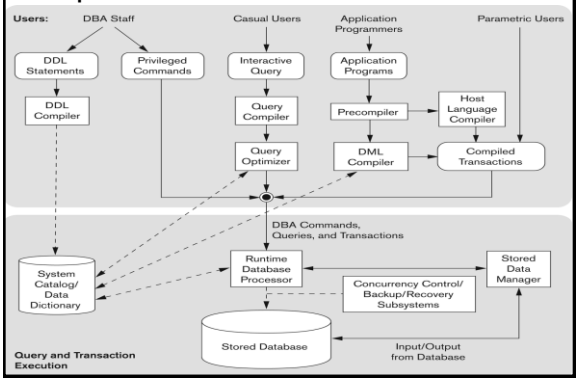
04. Data Access and Manipulation

Introduction: What is SQL

- Expanded: Structured Query Language
  - Initially called SEQUEL (Structured English Query Language)
- A special purpose programming language for managing database management systems
- Both a declarative and procedural language
- SQL considered a major reason for the success of relational databases:
  - It became a standard for relational databases => fewer migration worries
  - Users can write statements in a database application program that accesses two or more databases without having to alter language
- There still are differences in various dialects of SQL … but if users stick to the 'Standard SQL' and relational systems faithfully support it

Component Modules of a DBMS



Data Definition and Data Types

- Keywords: CREATE, DROP, and ALTER the descriptions of the tables (relations) of a database
- Required: descriptor of structure of table to be created (Data Types)
- Typically the role of the DBA and any other user explicitly authorized by DBA

Standard SQL Datatypes

- Character String:
  - CHAR(N), CHARACTER(N)
  - VARCHAR(n)
- Numeric:
  - INTEGER (OR INT), SMALLINT,
  - FLOAT (OR REAL), DOUBLE PRECISION
  - DECIMAL(I,J)
- Bit-string
  - BIT(n) – for fixed length
  - BIT VARYING(n) – n is the max
- BOOLEAN – values True, False and Unknown (in cases of 3-valued logic)
- DATE – handles values of format YYYY-MM-DD
- TIME – HH:MM:SS
- TIMESTAMP – Includes both date and time

CREATE SCHEMA

- Specifies a new database schema by giving it a name
- Database Schema = a set of relation schemas

CREATE TABLE

- Specifies a new relation by giving it a name, and specifying each of its attributes and their data types (INTEGER, FLOAT, DECIMAL(i,j), CHAR(n), VARCHAR(n))
- A constraint NOT NULL may be specified on an attribute

```
CREATE TABLE   DEPARTMENT
            (DNAME         VARCHAR(10)  NOT NULL,
            DNUMBER        INTEGER          NOT NULL,
            MGRSSN         CHAR(9),
            MGRSTARTDATE         Date);
```

7

CREATE TABLE- Additionals

- CREATE TABLE command for specifying the primary key attributes, secondary keys, and referential integrity constraints (foreign keys).
- Key attributes can be specified via the PRIMARY KEY and UNIQUE phrases

```
CREATE TABLE   DEPT
( DNAME                  VARCHAR(10)      NOT NULL,
   DNUMBER               INTEGER  NOT NULL,
   MGRSSN                CHAR(9),
   MGRSTARTDATE    Date,
   PRIMARY KEY (DNUMBER),
   UNIQUE (DNAME),
   FOREIGN KEY (MGRSSN) REFERENCES EMP(SSN)  );
```

8

DROP TABLE

- Used to remove a relation (base table) *and its definition*
- The relation can no longer be used in queries, updates, or any other commands since its description no longer exists
- Example:

DROP TABLE DEPENDENT;

9

ALTER TABLE

- Used to add an attribute to one of the base relations
- The new attribute will have NULLs in all the tuples of the relation right after the command is executed; hence, the NOT NULL constraint is *not allowed* for such an attribute
- Examples:
  - **ALTER TABLE Employee ADD Job VARCHAR(12);**
  - **ALTER TABLE Employee DROP Job**
- The database users must still enter a value for the new attribute JOB for each EMPLOYEE tuple. This can be done using the UPDATE command.

10

REFERENTIAL INTEGRITY OPTIONS

- We can specify RESTRICT, CASCADE, SET NULL or SET DEFAULT on referential integrity constraints (foreign keys)

```
CREATE TABLE   DEPT
(    DNAME        VARCHAR(10)        NOT NULL,
     DNUMBER   INTEGER      NOT NULL,
     MGRSSN     CHAR(9),
     MGRSTARTDATE  CHAR(9),
     PRIMARY KEY (DNUMBER),
     UNIQUE (DNAME),
     FOREIGN KEY (MGRSSN) REFERENCES EMP
ON DELETE SET DEFAULT ON UPDATE CASCADE  );
```

11

Retrieval Queries in SQL

- SQL has one basic statement for retrieving information from a database; the SELECT statement
- Important distinction between SQL and the formal relational model; SQL allows a table (relation) to have two or more tuples that are identical in all their attribute values
- SQL relations can be constrained to be sets by specifying PRIMARY KEY or UNIQUE attributes, or by using the DISTINCT option in a query

12

Retrieval Queries in SQL

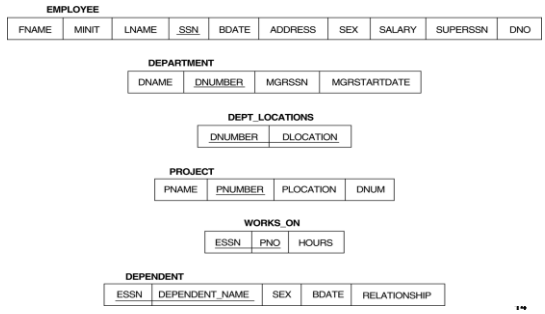
- Basic form of the SQL SELECT statement is called a *mapping* or a *SELECT-FROM-WHERE block*

SELECT <attribute list>
FROM <table list>
WHERE <condition>

- <attribute list> is a list of attribute names whose values are to be retrieved by the query
- <table list> is a list of the relation names required to process the query
- <condition> is a conditional (Boolean) expression that identifies the tuples to be retrieved by the query

13

Relational Database Schema



14

Specifying Updates in SQL

- There are three SQL commands to modify the database state; INSERT, DELETE, and UPDATE

15

INSERT

- In its simplest form, it is used to add one or more tuples to a relation
- Attribute values should be listed in the same order as the attributes were specified in the CREATE TABLE command

16

INSERT (cont.)

- **Example U1:**

INSERT INTO Employee
VALUES ('Richard', 'K', 'Waweru', '653298653', '12 Dec 1962', 'P O Box 98 Thika', 'M', 37000, '987654321', '4');

17

Alternative INSERT

- Specifies explicitly the attribute names that correspond to the values in the new tuple
- Attributes with NULL values can be left out
- Example: Insert a tuple for a new EMPLOYEE for whom we only know the FNAME, LNAME, and SSN attributes.

U1a:

INSERT INTO Employee (FName, LName, DNo, SSN)
VALUES ('Patrick', 'Mwachiro', '4', '653298654');

18

INSERT

- Important Note: Only the constraints specified in the DDL commands are automatically enforced by the DBMS when updates are applied to the database
- Another variation of INSERT allows insertion of *multiple tuples* resulting from a query into a relation

- Example: Suppose there's need to create a temporary table that has the name, number of employees, and total salaries for each department. A table DEPTS_INFO is created by U3A, and is loaded with the summary information retrieved from the database by the query in U3B.

19

INSERT

U3a:

```
CREATE TABLE Depts_Info
(Dept_Name VARCHAR(15), No_Of_Emps  Integer,
   Total_Sal  Integer );
```

U3b:

```
INSERT INTO Depts_Info ( Dept_Name, No_Of_Emps,
   Total_Sal )
SELECT DNAME, Count(*), Sum(Employee.Salary)
FROM Department, Employee
WHERE DNUMBER =DNo
GROUP BY DNAME;
```

20

DELETE

- Removes tuples from a relation
- Includes a WHERE-clause to select the tuples to be deleted
- Tuples are deleted from only *one table* at a time (unless CASCADE is specified on a referential integrity constraint)
- A missing WHERE-clause specifies that *all tuples* in the relation are to be deleted; the table then becomes an empty table
- The number of tuples deleted depends on the number of tuples in the relation that satisfy the WHERE-clause
- Referential integrity should be enforced

21

Delete

U4a:

```
DELETE *
FROM Employee
WHERE LName='Waweru';
```

U4b:

```
DELETE *
FROM Emp1
WHERE SSN='123456789';
```

U4c:

```
DELETE *
FROM Emp3
WHERE DNo IN
(Select DNumber From Department Where DName='Research');
```

U4d:

```
DELETE *
FROM Emp4;
```

22

UPDATE

- Used to modify attribute values of one or more selected tuples
- A WHERE-clause selects the tuples to be modified
- An additional SET-clause specifies the attributes to be modified and their new values
- Each command modifies tuples *in the same relation*
- Referential integrity should be enforced

23

UPDATE (cont.)

- Example: Change the location and controlling department number of project number 10 to 'Garissa' and 5, respectively.

U5:

```
UPDATE Proj SET PLocation = 'Garissa', Dnum
   = 5
WHERE PNumber=10;
```

24

UPDATE

Example (U6): Give all employees in the 'Research' department a 10% raise in salary.

```
UPDATE Emp2 SET Salary = Salary*1.1
WHERE DNO IN
        (SELECT DNumber
         FROM DEPARTMENT
         WHERE DName = 'Research');
```

- In this request, the modified SALARY value depends on the original SALARY value in each tuple

25

Populated Database

EMPLOYEE

FNAME	MINIT	LNAME	SSN	BDATE	ADDRESS	SEX	SALARY	SUPERSSN	DNO
John	B	Smith	123456789	1965-01-09	731 Fondren, Houston, TX	M	30000	333445555	5
Franklin	T	Wong	333445555	1955-12-08	638 Voss, Houston, TX	M	40000	888665555	5
Alicia	J	Zelaya	999887777	1968-07-19	3321 Castle, Spring, TX	F	25000	987654321	4
Jennifer	S	Wallace	987654321	1941-06-20	291 Berry, Bellaire, TX	F	43000	888665555	4
Ramesh	K	Narayan	666884444	1962-09-15	975 Fire Oak, Humble, TX	M	38000	333445555	5
Joyce	A	English	453453453	1972-07-31	5631 Rice, Houston, TX	F	25000	333445555	5
Ahmad	V	Jabbar	987987987	1969-03-29	980 Dallas, Houston, TX	M	25000	987654321	4
James	E	Borg	888665555	1937-11-10	450 Stone, Houston, TX	M	55000	null	1

DEPARTMENT

DNAME	DNUMBER	MGRSSN	MGRSTARTDATE
Research	5	333445555	1988-05-22
Administration	4	987654321	1995-01-01
Headquarters	1	888665555	1981-06-19

DEPT_LOCATIONS

DNUMBER	DLOCATION
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

ESSN	PNO	HOURS
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	null

PROJECT

PNAME	PNUMBER	PLOCATION	DNUM
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

DEPENDENT

ESSN	DEPENDENT_NAME	SEX	BDATE	RELATIONSHIP
333445555	Alice	F	1986-04-05	DAUGHTER
333445555	Theodore	M	1983-10-25	SON
333445555	Joy	F	1958-05-03	SPOUSE
987654321	Abner	M	1942-02-28	SPOUSE
123456789	Michael	M	1988-01-04	SON
123456789	Alice	F	1988-12-30	DAUGHTER
123456789	Elizabeth	F	1967-05-05	SPOUSE

Simple SQL Queries

- Basic SQL queries correspond to using the SELECT, PROJECT, and JOIN operations of the relational algebra
- All subsequent examples use the COMPANY database
- Example of a simple query on *one* relation
- Query 0: Retrieve the birthdate and address of the employee whose name is 'John'.

```
Q0:SELECT   BDATE, ADDRESS
   FROM          EMPLOYEE
   WHERE    FNAME='John'
```

27

Simple SQL Queries

- Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

```
Q1
SELECT FName, LName, Address
FROM Employee, Department
WHERE DName='Research' And DNumber=DNo;
```

- Similar to a SELECT-PROJECT-JOIN sequence of relational algebra operations
- (DNAME='Research') is a *selection condition* (corresponds to a SELECT operation in relational algebra)
- (DNUMBER=DNO) is a *join condition* (corresponds to a JOIN operation in relational algebra)

28

Simple SQL Queries (cont.)

- Query 2: For every project located in 'Naivasha', list the project number, the controlling department number, and the department manager's last name, address, and birthdate.

```
SELECT PNumber, DNum, LName, BDate, Address
FROM Project, Department, Employee
WHERE DNum=DNumber AND MgrSSN=SSN AND PLOCATION='Naivasha';
```

- The join condition **DNUM=DNUMBER** relates a project to its controlling department
- The join condition **MGRSSN=SSN** relates the controlling department to the employee who manages that department

29

Aliases, * and DISTINCT, Empty WHERE-clause

- In SQL, we can use the same name for two (or more) attributes as long as the attributes are in *different relations*
- A query that refers to two or more attributes with the same name must *qualify* the attribute name with the relation name by *prefixing* the relation name to the attribute name

Example:

- EMPLOYEE.NAME, DEPARTMENT.NAME

30

ALIASES

- Some queries need to refer to the same relation twice
- In this case, *aliases* are given to the relation name
- Query 8: For each employee, retrieve the employee's name, and the name of his or her immediate supervisor.

Q8:

```
SELECT E.Fname, E.LName, S.FName, S.Lname
FROM Employee E, Employee S
WHERE E.SuperSSN=S.SSN;
```

- The alternate relation names E and S are called *aliases* for the EMPLOYEE relation
- We can think of E and S as two *different copies* of EMPLOYEE; E represents employees in role of *supervisees* and S represents employees in role of *supervisors*

31

ALIASES (cont.)

Aliasing can also be used in any SQL query for convenience. Can also use the AS keyword to specify aliases

```
SELECT E.Fname, E.LName, S.FName, S.Lname
FROM Employee AS E, Employee AS S
WHERE E.SuperSSN=S.SSN;
```

32

UNSPECIFIED WHERE-clause

- A *missing WHERE-clause* indicates no condition; hence, *all tuples* of the relations in the FROM-clause are selected
- This is equivalent to the condition WHERE TRUE
- Query 9: Retrieve the SSN values for all employees.

```
Q9:   SELECT   SSN
      FROM EMPLOYEE
```

- If more than one relation is specified in the FROM-clause *and* there is no join condition, then the *CARTESIAN PRODUCT* of tuples is selected

Q10:

```
SELECT SSN, DName
FROM Employee, Department;
```

33

USE OF *

- To retrieve all the attribute values of the selected tuples, a * is used, which stands for *all the attributes*

Examples:

```
Q1C:  SELECT  *
      FROM    EMPLOYEE
      WHERE   DNO=5

Q1D:  SELECT  *
      FROM    EMPLOYEE, DEPARTMENT
      WHERE   DNAME='Research' AND
              DNO=DNUMBER
```

34

USE OF DISTINCT

- SQL does not treat a relation as a set; *duplicate tuples can appear*
- To eliminate duplicate tuples in a query result, the keyword **DISTINCT** is used
- For example, the result of Q11 may have duplicate SALARY values whereas Q11A does not have any duplicate values

Q11:

```
SELECT SALARY FROM EMPLOYEE
```

Q11A:

```
SELECT DISTINCT SALARY FROM EMPLOYEE
```

35

SET OPERATIONS

- SQL has directly incorporated some set operations
- Operations:
  - Union
  - Minus - Difference
  - Intersect
- The resulting relations of these set operations are sets of tuples; *duplicate tuples are eliminated from the result*
- The set operations apply only to *union compatible relations* ; the two relations must have the same attributes and the attributes must appear in the same order

36

SET OPERATIONS

- Query 4: Make a list of all project numbers for projects that involve an employee whose last name is 'Omondi' as a worker or as a manager of the department that controls the project.

Q4:

```
(SELECT PName
FROM Project, Department, Employee
WHERE DNUM=DNumber AND MGRSSN=SSN AND LNAME='Omondi')
UNION
(SELECT PName
FROM Project, Works_On, Employee
WHERE PNumber=PNo AND ESSN=SSN AND LName='Omondi');')
```

37

NESTING OF QUERIES

- A complete SELECT query, called a *nested query* , can be specified within the WHERE-clause of another query, called the *outer query*
- Many of the previous queries can be specified in an alternative form using nesting
- Query 1: Retrieve the name and address of all employees who work for the 'Research' department.

Q1 - Nested

```
SELECT FNAME, LNAME, ADDRESS
FROM EMPLOYEE
WHERE DNO IN
        (SELECT DNUMBER
          FROM DEPARTMENT
          WHERE DNAME='Research')
```

38

NESTING OF QUERIES

- The nested query selects the number of the 'Research' department
- The outer query select an EMPLOYEE tuple if its DNO value is in the result of either nested query
- In general, we can have several levels of nested queries

39

EXPLICIT SETS|

- It is also possible to use an **explicit (enumerated) set of values** in the WHERE-clause rather than a nested query
- Query 13: Retrieve the social security numbers of all employees who work on project number 1, 2, or 3.

```
Q13:    SELECT    DISTINCT ESSN
        FROM      WORKS_ON
        WHERE     PNO IN  (1, 2, 3)
```

40

SUBSTRING COMPARISON

- The **LIKE** comparison operator is used to compare partial strings
- Two reserved characters are used: '%' (or '*' in some implementations) replaces an arbitrary number of characters, and '_' replaces a single arbitrary character

41

SUBSTRING COMPARISON

- Query 12: Retrieve all employees whose address is in Nairobi. Here, the value of the ADDRESS attribute must contain the substring 'Nairobi'.

Q12:

```
SELECT FName, LName, Address
FROM Employee
WHERE Address LIKE '*Nairobi*';
```

42

ARITHMETIC OPERATIONS

- The standard arithmetic operators '+', '-'. '*', and '/' (for addition, subtraction, multiplication, and division, respectively) can be applied to numeric values in an SQL query result
- Query 13: Show the effect of giving all employees who work on the 'ProductX' project a 10% raise.

Q13:

```
SELECT FNAME, LNAME, 1.1*SALARY
FROM EMPLOYEE, WORKS_ON, PROJECT
WHERE SSN=ESSN AND PNO=PNUMBER AND
   PNAME='ProductX'
```

43

Ordering Query Results

- The **ORDER BY** clause is used to sort the tuples in a query result based on the values of some attribute(s)
- Query 15: Retrieve a list of employees and the projects each works in, ordered by the employee's department, and within each department ordered alphabetically by employee last name.

Q15:

```
SELECT DName, LName, FName, PName
FROM Department, Employee, Works_On, Project
WHERE DNumber=DNo AND SSN=ESSN AND
   PNO=PNumber
ORDER BY DName, LName, FName;
```

44

Ordering Query Results

- The default order is in ascending order of values
- To specify use the keywords:
  - **DESC** for descending order;
  - **ASC** for ascending order, though it is the default

45

NULLS IN SQL QUERIES

- SQL allows queries that check if a value is NULL (missing or undefined or not applicable)
- SQL uses **IS** or **IS NOT** to compare NULLs because it considers each NULL value distinct from other NULL values, so equality comparison is not appropriate .
- Query 14: Retrieve the names of all employees who do not have supervisors.

Q14:

```
SELECT FNAME, LNAME
FROM EMPLOYEE
WHERE SUPERSSN  IS  NULL
```

Note: If a join condition is specified, tuples with NULL values for the join attributes are not included in the result

46

AGGREGATE FUNCTIONS

- Include **COUNT**, **SUM**, **MAX**, **MIN**, and **AVG**
- Query : Find the sum of salaries, maximum salary, the minimum salary, and the average salary among all employees.

Q19:

```
SELECT SUM(Salary), MAX(Salary), MIN(Salary), AVG(Salary)
FROMEMPLOYEE
```

47

AGGREGATE FUNCTIONS (cont.)

- Query 20: Find the maximum salary, the minimum salary, and the average salary among employees who work for the 'Research' department.

Q20:

```
SELECT MAX(SALARY), MIN(SALARY), AVG(SALARY)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research'
```

48

AGGREGATE FUNCTIONS (cont.)

- Queries 21 and 22: Retrieve the total number of employees in the company (Q21), and the number of employees in the 'Research' department (Q22).

Q21:

```
SELECT COUNT (*) FROM EMPLOYEE
```

Q22:

```
SELECT COUNT (*)
FROM EMPLOYEE, DEPARTMENT
WHERE DNO=DNUMBER AND DNAME='Research'
```


GROUPING

- In many cases, we want to apply the aggregate functions *to subgroups of tuples in a relation*
- Each subgroup of tuples consists of the set of tuples that have *the same value* for the *grouping attribute(s)*
- The function is applied to each subgroup independently
- SQL has a **GROUP BY**-clause for specifying the grouping attributes, which *must also appear in the SELECT-clause*

GROUPING

- Query 24: For each department, retrieve the department number, the number of employees in the department, and their average salary.

Q24:

```
SELECT DNO, COUNT (*), AVG (SALARY)
FROM EMPLOYEE
GROUP BY  DNO
```

- In Q20, the EMPLOYEE tuples are divided into groups – each group having the same value for the grouping attribute DNO
- The COUNT and AVG functions are applied to each such group of tuples separately
- The SELECT-clause includes only the grouping attribute and the functions to be applied on each group of tuples
- A join condition can be used in conjunction with grouping

GROUPING

- Query 25: For each project, retrieve the project number, project name, and the number of employees who work on that project.

Q25:

```
SELECT PNumber, PName, Count(*) AS [Number of
  Workers]
FROM Project, Works_On
WHERE PNumber=PNO
GROUP BY PNumber, PName
```

- In this case, the grouping and functions are applied *after* the joining of the two relations

Summary of SQL Queries

- A query in SQL can consist of up to six clauses, but only the first two, SELECT and FROM, are mandatory. The clauses are specified in the following order:

```
SELECT      <attribute list>
FROM   <table list>
[WHERE      <condition>]
[GROUP BY <grouping attribute(s)>]
[HAVING     <group condition>]
[ORDER BY <attribute list>]
```


Summary of SQL Queries

- The SELECT-clause lists the attributes or functions to be retrieved
- The FROM-clause specifies all relations (or aliases) needed in the query but not those needed in nested queries
- The WHERE-clause specifies the conditions for selection and join of tuples from the relations specified in the FROM-clause
- GROUP BY specifies grouping attributes
- HAVING specifies a condition for selection of groups
- ORDER BY specifies an order for displaying the result of a query
- A query is evaluated by first applying the WHERE-clause, then GROUP BY and HAVING, and finally the SELECT-clause